LEVEL II ②

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
SELECTE
JAN 9 1981
E

THESIS

IMPLEMENTATION OF THE GRAPHICS-ORIENTED
INTERACTIVE FINITE ELEMENT TIME-SHARING
SYSTEM (GIFTS) ON THE PDP-11.

by

John Trevor Sheldon

September 1980

Thesis Advisor: Gilles Cantin

81 1   08 032

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A093 597	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Implementation of the Graphics-oriented Interactive Finite Element Time-sharing System (GIFTS) on the PDP-11		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1980
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John Trevor Sheldon		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1980
		13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

GIFTS	Graphics
Finite Element	Minicomputer
Structural Analysis	PDP-11

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Graphics-oriented, Interactive, Finite Element Time-sharing System (GIFTS), written by Professor A. Kamel and Mr. Michael McCabe of the University of Arizona, has been implemented on the PDP-11 at the Naval Postgraduate School. This powerful system of programs was installed in a manner to facilitate its modification in the future. A very brief description of the GIFTS system, including the Unified Data Base, as well as the PDP-11 and RSX-11M

20. ABSTRACT (continued)

operating system, are provided. Finally, a systematic approach
to building and/or modifying the GIFTS system in the future is
included. The approach taken includes a "File Sorter" program
which removes the need for much of the tedious work associated
with building the system.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

2

Implementation of the Graphics-oriented Interactive
Finite Element Time-sharing System (GIFTS) on the PDP-11

by

John Trevor Sheldon
Lieutenant Commander, United States Navy
B. S., United States Naval Academy, 1967
M. S., Naval Postgraduate School, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1980

Author _____

Approved by: _____

Thesis Advisor

_____

Second Reader

_____

Chairman, Department of Mechanical Engineering

_____

Dean of Science and Engineering

3

ABSTRACT

The Graphics-oriented, Interactive, Finite Element
Time-sharing System (GIFTS), written by Professor A. Kamel
and Mr. Michael W. McCabe of the University of Arizona, has
been implemented on the PDP-11 at the Naval Postgraduate
School. This powerful system of programs was installed in
a manner to facilitate its modification in the future. A
very brief description of the GIFTS system, including the
Unified Data Base, as well as the PDP-11 and RSX-11M operating
system, are provided. Finally, a systematic approach to
building and/or modifying the GIFTS system in the future is
included. The approach taken includes a "File Sorter" pro-
gram which removes the need for much of the tedious work
associated with building the system.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to take this opportunity to thank
Professor Gilles Cantin for his patience and assistance,
and Professor Paul Marto who, in Professor Cantin's
absence, helped me through a major crisis. Also, there are
not words to describe the patience shown by my family
through my two tours at the Naval Postgraduate School.
They deserve a lot more than merely mentioning their names,
but notwithstanding this: Alice, Patricia, John and Peter.

I. INTRODUCTION

The purpose of this thesis is to describe the process
whereby the Graphics-oriented Interactive Finite Element
Time-sharing System (GIFTS) was implemented at the Naval
Postgraduate School.

Anyone who has entered a problem with a large amount
of numerical input into a computer knows the fear of making
logical or typing errors which will go undetected. The
GIFTS system goes a long way in reducing this problem by
allowing a user to graphically reproduce the problem he/she
has entered into the system. The solved problem can be
displayed, as well, in a form which makes the effect of a
given loading graphically reproducible at any time in the
future.

The first step in building the system was obtaining the
latest version (5.02) of the taped program from the Inter-
active Graphics Engineering Laboratory (IGEL) of the
University of Arizona. After an initial attempt at "building"
the system on the PDP-11 (using the methods described below)
several minor errors were found. These errors were generally
in the form of incomplete revisions and were easily correct-
able with telephone assistance from one of the developers,
Mr. Michael W. McCabe, of the University of Arizona.

Since the average mechanical engineering student at the Naval Postgraduate School does not ordinarily spend time being taught on a computer system other than the school's mainframe (currently the IBM 360/67), a great deal of time was required in the preparation for this thesis simply learning the PDP-11/50 and the RSX-11M Operating System. Since it is expected that the GIFTS system will need to be revised in the future, it became obvious that an important objective of this thesis was to develop the means whereby changes to GIFTS could be made as conveniently and "painlessly" as possible without the need for a student or faculty member to become intimately familiar with the PDP-11. It is believed that this objective has been successfully met with the combination of a File Sorter program (FILSOR) and two "command" files. The net impact of these three programs is to allow a most powerful Finite Element Method (FEM) pre and post processer (plus solver) to be completely built on the PDP-11 with two tapes, two commands, and six hours of time.

It is believed that the GIFTS system will, in the future, be an invaluable teaching aid at the Naval Postgraduate School.

II. DESCRIPTION OF GIFTS

GIFTS is a system of programs used in solving Finite Element Problems. This statement does not really do justice to the system for the forte of the system is not in its ability to mathematically solve problems but rather in its ability to reliably and fairly completely define structural problems which are to be solved. It allows a user to:

1) Input problem parameters;

2) Observe the input both graphically and in tabulated form;

3) Update the model; and

4) Observe the output

Many problems, due to their sizes, will be outside the range of the "solver" contained in the program. But, due to the highly structured nature of the Unified Data Base (UDB), other systems, more powerful in this area, can access the data, solve the problem, and return the solved problem to GIFTS for display.

The purpose of this section is to give enough of a description of GIFTS and the available documentation to assist a user interested in solving a FEM problem to find out how to get started at the Naval Postgraduate School.

A. GIFTS DEVELOPERS

GIFTS was written by Professor Hussein A. Kamel and
Mr. Michael W. McCabe of the University of Arizona. The
system is constantly being revised/updated as the need
arises. The facility for expansion of the system is built
in to it as not all element types have beem implemented.
As updates are received, they can be implemented by the
procedures outlined below in section V.

B. SYSTEM CAPABILITIES

Much of the information included in this section is
already included, in substance, in the "GIFTS Users'
Manual." It is the purpose here to synthesize the informa-
tion from this reference needed to have a general under-
standing of the system.

A list of the several program modules with descriptions
can be found in Appendix A. Each has a purpose in formulating
a FEM problem and more than one module is necessary to
completely formulate a problem. However, not all program
modules are necessary for every formulation.

The general breakdown of the module types/purposes is:

1) Model Generation and Editing;

2) Load and Boundary Conditions Generation, Display
and Editing; and

3) General Purpose Computational and Result Display
Modules.

12

In addition, there are modules available (but not yet
implemented at the Naval Postgraduate School) to inter-
face the GIFTS system with other Finite Element programs
including:

1) ANSYS

2) SAP4

3) NASTRAN

The purpose of these interfaces is to act as "interpreters"
of the GIFTS Unified Data Base in order that the generated
model may be solved on one of these other systems. The
interface program also takes the solutions generated by the
other system and formats them back into the UDB for GIFTS
in order that they can be displayed.

In the "GIFTS Users' Reference Manual," it is stated:
"the generation and display capabilities of GIFTS go beyond
its own analysis capabilities." It gives, by example, the
fact that the GIFTS system can generate and display higher
order elements while not (yet) being able to analyze the
results. Though the author is not privy to a timetable, it
is expected that the system capabilities will increase and
can be added to existing capabilities currently at the Naval
Postgraduate School. The methodology for making such revi-
sions is covered below in chapter V.

1. Pre and Post Processing Capabilities

    a. Pre Processing

        The GIFTS system is capable of being used as a
pre-processor for other systems. It accepts commands which
allow a user to establish the geometry of a model and to
display it at a terminal for verification.

        Figure 1 is an example of the program/user
interaction which is required to establish the geometry of
a plate with a hole in the center.[1] Figure 2 is the resulting
plot with elements and points labelled. Should an error be
made during the session, a correction can be made before
going on.

        Also available to the user are a variety of
tabulations of input and computed data. These also prove
useful in the verification of a model.

    b. Post Processing

        Figure 3 depicts the results of the solved FEM
problem which was entered as in section 1.a. above. It
depicts the stress contours as computed by (in this case)
the GIFTS system for a given loading. If a different
solver (e.g. SAP4) were to be used, the interface program
would "translate" the output from the solver into the

---

        [1]This problem is one that was included in the "GIFTS
Primer" which was written at IGEL, University of Arizona.
See Appendix C.

GIFTS UDB format before using the GIFTS modules for displaying the results.

The system can also display deflection plots due to a given loading as well as computing and displaying time domain problems.

2. Solving Capabilities

The system, as presently configured, is capable of solving a wide class of structural, finite element method problems. However, there are some limitations. Page III-1 of the GIFTS User's Manual lists those elements which enjoy "Full Support" and, also, those within the categories: "Generation and Display Only" and "Storage Only." A user should be aware of these distinctions before deciding to solve a problem completely by GIFTS or by GIFTS in conjunction with another system.

C. THE COMPUTER PROGRAM

If loaded all at once, with no overlaying, the entire set of program/modules would take up to perhaps two mega bytes of memory. Since it is not desirable (or usually possible) to have this much space available to a user, the program has been divided into several, separately executable modules having as their common denominator the Unified Data Base.

Each program is called up (executed) by a "RUN" command. At the end of the session with an interactive module, a

"QUIT" (or similar) command is given which causes the
module to update the data base, close files and leave the
module. To enter the next module another "RUN" command is
given and so forth.

The interactive modules accept a large number of well-
defined commands. Some of the modules have similar and
even duplicative sub-objectives and therefore contain
many of the same commands. Each program, however, has its
own communications subroutine which will accept commands
only valid in the particular module. Several different type
prompt symbols are used (>, *, ?, blank) which make the
nature of the input (i.e. alphanumeric, integer or floating
point) less ambiguous.

As it was earlier stated, overlaying is required for
most modules when installed on the PDP-11. This is due to
a 64K byte limitation for any program segment. The overlaying
schemes used for the several modules were included on the
tapes received from the IGDL, University of Arizona, and
are duplicated on the tapes discussed below in section IV.C.

One of the capabilities available to the user in many
modules is that of plotting the model at a terminal. This
feature obviously requires that a terminal with a graphics
capability be used. The terminal for which the GIFTS
System at the Naval Postgraduate School (NPS) is presently

geared is the Tektronix 4000 Series. To change to another
type of terminal would require modifications to several of
the GIFTS library subroutines.[2]

D. THE UNIFIED DATA BASE

During the course of model definition, the GIFTS system
opens, performs input/output (I/O), and closes files on
disc. At the end of a session one will find on his/her
disc space several files having the jobname (specified
by the user) as their filenames but each having a different
"extension." These files represent the Unified Data
Base (UDB).

The UDB files created by GIFTS are primarily random
access, unformatted disc files. The fact they are unformatted
makes storage of numerical data more efficient and the
random access feature allows for easier identification of
a particular record to be read or written.

1. Definition of Terms

The individual files are described in the "GIFTS
Systems Manual" but a general description of the methodology
of the programmers and the terminology used in the manual
is warranted.

A "physical record," in context with the terminology
used in the Systems Manual is the "collection of data" found

_____

[2]Specifically those in the file: MITS.PDP

17

in one record. When submitting a program written on punched cards, the input record length is limited to 80 - the number of characters allowed on the card. A disc, of course, does not have this limitation and the record size can be extremely large. (In the GIFTS system, the record size is automatically defined within the program and can be as large as 1684 bytes.) The program uses equivalent size buffers to accomodate the I/O record sizes and also allows for variable sizing of buffers/record size should the programs be run on a large machine. This fact is academic though since the current installation at the Naval Postgraduate School is on the PDP-11.

A "logical record" is a term used for the grouping together of data which are related insofar as the programmer says they are. Better put: "the smallest collection of data into which the data contained in a file may be divided." For example, a physical record of 200 bytes could be divided into ten logical records of 20 bytes each. In this case, the number "10" is the "blocking factor."

Data in GIFTS are generally buffered in named COMMON. A buffer typically consists of a physical record plus some "bookkeeping" data. For example, a physical record in the "points" (PTS) file contains data on ten points. If the point being worked on by GIFTS is not in the record currently in the buffer, the current buffer is

stored in the PTS file and the correct record is read in.
The "logical record" needed by GIFTS (i.e. the point being
worked on) is now available.

2. Naming Convention

The UDB consists of, typically, ten or so files
(the exact number depending on the problem being modeled).
Certain administrative files are kept open throughout the
life of a problem - that is, until deleted by the user.
These files do not contain data which are used directly in
solving or displaying a specific model. Other files are
temporary or "scratch" files and are deleted prior to
leaving the module which opened them. Then there are the
files containing the data unique to a model which are
"passed on" from module to module until the model/solution
is completed. All of these files are the Unified Data
Base - the focal point of the GIFTS system.

At the beginning of each module, the user is queried
concerning the name of the model. The first time that this
name is used (usually in the modules BUILT or EDIT), the
name becomes unique until the problem is deleted from the
disc.[3]

Figure 4 is a sample listing of the files built by
GIFTS for the job "PLATE" which was shown in previous

---

[3]This cannot be done by GIFTS but must be done with the
file handler - see section III.C.3.

section. As the problem progresses, the number of files
could increase and eventually take up a great deal of
disc space (users should keep this in mind when creating a
problem when disc space is at a premium).

Two other files exist which do not follow the
naming convention which was outlined above. These are:
GIFTS5.INF and GIFTS5.EST. The former is a sequential,
formatted file listing all the "HELP" command answers that
are available. It requires updating as changes are made
to the system and is not tied to any particular problem.
The latter file is used by OPTIM (i.e. optimization program)
and is strictly for time estimates for the problem being
completed. The user normally need not be concerned with
this file, as it should already exist. If it doesn't, this
will cause minor problems and could be easily rebuilt by
running the module EST.TSK which is included on the magnetic
tapes discussed below in section IV.G.

Each of the files is described in the GIFTS System
Manual beginning on page SM II-2. For those interested in
modifying the GIFTS system or writing an interface program,
further explanation of the UDF is given below in section IV.

E. DOCUMENTATION OF GIFTS

The source listing as provided by the IGEL, University
of Arizona, is liberally filled with comment statements.
However, the interaction of the approximately 300 library

subroutines with the program and subroutines within the modules is so complex that trying to understand exactly what a program is doing at a particular time is virtually impossible without an excessive expenditure of time.

The user normally will not be interested in the source listing but rather in how to RUN the system. The remainder of this section is devoted to the documentation provided by the developers on how to use the system to solve a problem.

1. Reference Manuals

There are several manuals which are of interest to both the GIFTS user and the systems analyst responsible for building or maintaining GIFTS (see Appendix B). The manuals are provided with the GIFTS system by the University of Arizona and are kept at the Naval Postgraduate School in the Mechanical Engineering Department Computer Laboratory, Room 201D, Halligan Hall.[4]

Two of these manuals have already been mentioned above: "The GIFTS Systems Manual"; and the "GIFTS User's Reference Manual." The former was used extensively in attempting to understand how the computer program worked and to understand the UDB. The latter was used in conjunction

---

[4]Not all the manuals have been provided by IGEL, University of Arizona, as they are yet to be written. For example, the "GIFTS System Installation Manual," which would have been useful here, has not yet been completed.

21

with the "GIFTS Primer" to obtain an understanding of how
the system operated from the user's viewpoint.

The "Primer" serves as an excellent aid for the
cautious, first-time user to get some hands-on experience
with the system and to see what the system can actually do.
It also explains, in detail, the purpose of several of the
commands. The included examples, besides being educational
for the first-time user, are very useful in checking the
installation for accuracy.

III. THE PDP-11

The GIFTS system has been installed on the PDP-11/50, located in Room 500, Spanagel Hall, at the Naval Post-graduate School. The choice to install the system on this particular computer was based on its availability; the fact that GIFTS had already been brought up on the PDP-11 elsewhere; and, that the main computer system at NPS, the IBM 360, was being replaced in the near future.

There are actually two PDP-11s available in the Computer Lab: one with the UNIX operating system, and the other with RSX-11M. GIFTS was installed in the latter as it is limited to 32K work (64K byte) segments whereas UNIX allows only a 16K work (32K byte) segment size.

A. ORGANIZATION

The Computer Laboratory at the Naval Postgraduate School falls under the administrative control of the Director, Computer Laboratories. Under his/her control are several analysts/mathematicians familiar with the RSX-11M operating system.

B. RSX-11M OPERATING SYSTEM

The following are descriptions of utilities available under RSX-11M and which are used in the building and running

of GIFTS. The descriptions are provided here primarily as background information for section IV. The details of the following utilities may be found in the appropriate PDP-11 manuals.

1. LOGON (HEL)

"HEL" is the logon keyword. For the Mechanical Engineering Department, the logon is "HEL MEDEPT" whereupon the computer queries the user for an appropriate password. The logoff "keyword" is simply "BYE."

2. User Identification Code (UIC)

The UIC is assigned by the Director, Computer Laboratories, and serves two primary purposes in the RSX-11M operating system:

    a. Identification of a particular user for security and accounting purposes; and

    b. Identification of the user's directory on disc.

3. Peripheral Interchange Program (PIP)

PIP is the very versatile system of file handlers which is used to: move, delete, copy, rename, etc., files created on disc.

Some knowledge of PIP is essential to any prospective user of the GIFTS modules on the PDP-11. It allows for deleting and transferring files - which are useful "housekeeping" functions to know.

4. File Transfer Program (FLX)

FLX is the PDP-11 utility for handling files between disc and magnetic tapes.

5. FORTRAN Four Plus (F4P)

The FORTRAN compiler used to build the GIFTS system. The syntax for this system allows for the use of many "switches." In the building of GIFTS on the PDP-11 it was only necessary to use two switches:

a. /CO:20 - This switch was necessary on several of the subroutines to increase the number of allowed continuation cards from the default (i.e. 5).

b. /TR:NONE - This switch was used to build a separate system library which did not include the code necessary for tracing in the event of an object time error. This omission is necessary to allow the two largest modules to fit into the 32K word allowable segment on the PDP-11. (This will be further explained in section IV below.)

6. Taskbuilder (TKB)

TKB is the "linker" used under RSX-11M. In conjunction with command files, it builds executable modules complete with overlays. A description of the command files used for building GIFTS is given in section IV. Further knowledge of the TKB utility would only be necessary if one were to rebuild or modify the GIFTS system without the help of the techniques which will be demonstrated in section V.

7. Librarian Utility Program (LBR)

This utility is used to create and modify libraries of files. In the case of the building of the GIFTS system, it is used to create "system" and "module" libraries. In modifying the GIFTS system one would only need to become familiar with the syntax of two "switches": /IN = (that is, "insert"); and /DE: ("delete").[5] Examples are shown in section V.

8. Text Editor (EDT)

The RSX-11M system at the Naval Postgraduate School offers two text editors. "EDT" was selected because of its power. With a little imagination, a great deal of time can be saved in making major and/or repetitive changes to a file with EDT. To make future revisions to GIFTS, it is quite obvious that a knowledge of an editor would be necessary.

9. Macro Assembler (MAC)

This is the keyword for assembling macro programs. For example, to assemble a program called TEST.MAC, one could enter:

                    MAC TEST = TEST

This would produce an object module called TEST. It is also possible to get a listing of the program with all external

---

[5]Note the syntax difference in the use of "=" for /IN and ":" for /DE.

references, etc. For details concerning this and other features, a user should refer to the appropriate PDP-11N manual.

10. Execution Order (RUN)

RUN is the command under RSX-11N which causes an executable module to be loaded and executed. For example: RUN BULKM. For files that are overlaid, the executable module (with a default file extension of TSK) will need an additional file with the extension "STB." This is the "symbol table file" which is also built at the time of taskbuilding.

11. Use of Command Files

"Command" files are ASCII formatted files having an extension of "CMD." A Command file is executed by simply inserting the character "@" before the filename. For example, to run a command file called GIFTS5.CMD, one would type in:

@GIFTS5

The contents of the file would be executed line by line.

Another way in which command files can be used is in conjunction with a utility or the FORTRAN compiler, F4P. For example, if there are two separate FORTRAN programs to

be compiled, TEST1.FTN and TEST2.FTN, one could edit a

command file called TEST1.CMD as follows:

```
>EDT TEST1.CMD
*I
TEST 1 = TEST 1
TEST 2 = TEST 2
{ctrl}Z                      6
*exit
To execute this command file, one types:

    F4P @TEST1
```

This method can also be used for: PIP, TKB and LBR.

---

[6]CTRL Z is the combination of characters which allows
the user to leave the "input mode" in EDT.

IV. THE BUILDING OF GIFTS ON THE PDP-11

The process of building GIFTS can be broken down into
a few logical steps:

1) Sorting

2) Compiling

3) Library Building

4) System Building

5) Cleanup

To simplify some of these time consuming processes which
must be completed, the author has written a "File Sorter"
program (FILSOR) which effectively reduces the "slave labor"
time and improves, it is believed, the accuracy of this
process.

A. SOURCE TAPE

The PDP-11 version of the GIFTS system arrived on an
unlabelled, ASCII-formatted, nine-track tape. Along with
the tape was a listing of the names of the files and the
sizes thereof. The files can be broken down by type as
follows:

Concatenated FORTRAN programs/subroutines:	29
Overlay Description Files:	15
Macro Programs	2
GIFTS Information File	1
Test Programs (FORTRAN)	3

29

The listings of FORTRAN programs/subroutines are
unusable in the form they are received and must be separated,
compiled, and the object code placed in libraries before
the taskbuilding (linking) process can even begin. The
steps that would be involved if this separation process
were to be done manually with the Text Editor (EDT) are:

1. Find the first line of the program, subroutine or
function; then

2. Find the last line of the program, subroutine or
function (i.e. "END"); then

3. Write the inclusive lines between the first and
last lines out to a new file; then

4. Go back to 1. until an EOF is reached.
The finding of the first and last lines using EDT is not
difficult (except in each case, one must look for either
a subroutine or a function since both occur). Writing to
a new file is not particularly difficult but requires a
rather lengthy line of commands. For example, to write
lines 10130 through 13450, inclusive to a new file named
FILNAM.FTN, requires:

                WR10130:13450/FI:FILNAM.FTN/TN
It can be imagined how long it would take to do this
hundreds of times (about six hundred for GIFTS) without
an error! For the reason that this task is so tedious
and fraught with peril, the author wrote FILSOR.

B. SORTING OF SOURCE LISTING

  1. Description of FILSOR

      The basic FILSOR program accepts as input, the name
of a source listing file[7] containing at least two subroutines
or one main program plus one or more subroutines (or functions).
The following restrictions or guidelines concerning the use
of FILSOR exist:

      a. There are no "Block Data" subroutines within
the source listing to be sorted;

      b. If a listing contains a main program (vice a
subroutine or a function), it must appear first in the file;

      c. In all cases, the sorted program, subroutines
and functions will have to be compiled;

      d. In some cases, entire systems of sorted subroutines
will have to be compiled with the "/TR:NONE" switch in use;

      e. In all cases, the sorted and compiled subroutines
will have to be stored in a library called, arbitrarily,
L1.OLB;

      f. Comment cards preceding the first executable
statement are discarded from the first output program;

----

      [7]The current version of the GIFTS source listings are
on a magnetic tape in a format useable under the FLX utility
of the RSX-11M operating system (see above, section III.C.4).
To obtain a listing of a particular magtape file, it would
be necessary to load the file onto disc using FLX and then
printing it using PIP.

g.   The input listing is unaffected by FILSOR;

h.   The "END" statement images must begin in
column seven (otherwise the program will fail to recognize
it as being the last statement in the program).

All the FORTRAN source listings included with
GIFTS conform to the above restrictions.[8]

The main output from FILSOR is, of course, the
separated FORTRAN files. The files are named according to
the subroutine/function name or, in the case of a main
program, the name of the input file. For example, assume
a file named EXAMPL.EXT contains: a main program and
three subroutines (subroutines TEXT1, TEXT2 and Function
TEXT3). The results of running EXAMPL.EXT through FILSOR
would be to create four new files called:

       EXAMPL.FTN
       TEXT1.FTN
       TEXT2.FTN
       TEXT3.FTN

The original file EXAMPL.EXT, containing the concatenated
FORTRAN files, still exists and is unchanged by running
FILSOR.

---

[3] It should be emphasized that in the first program
or subroutine in the file, blank or comment cards pre-
ceding the first executable statement will be "lost."
Similarily, blank or comment cards between "END" and the
next subroutine or function within the listing will be
lost. Both these statements apply only to the sorted
routines - not the original listing.

FILSOR also builds two additional files while sorting the input file. Since it will eventually be necessary to compile all the subroutines, a file called LIB.CMD (a command file) is built which allows for the compilation of all program/subroutines sorted by FILSOR. As stated in section III.B.5, two possible combinations of "switches" occurred in the building of GIFTS: one with the "/TR:NONE" switch, and one without. The "/CO:20" switch was used regardless. FILSOR queries the user in order to determine whether he/she wants to include the trace capability or not.

The other file built by FILSOR is called "STUFF.CMD." This file, in conjunction with the LBR utility, will store the object modules compiled with LIB.CMD into an object library called L1.OLB. After the "STUFF" process, the library L1.OLB can be renamed using PIP to avoid confusion with future FILSOR operations.

Appendix C is a complete listing of FILSOR. An example session is included in Appendix D.

A more complicated version of FILSOR which allows a user to setup an input file with a list of several input files to be sorted is also included on magnetic tape.[9]

---

[9]This version is called FILS2 and requires an additional program, STUFFE, to build the input file. Both of these modules are included on tape and are executed as a part of the automatic "BUILDT" command file discussed below in section V and listed in Appendix E.

C. THE OVERLAID TBS

Most GIFTS modules must be overlaid on the PDP-11 in
order that they can fit into memory. The syntax involved
with the taskbuilder is quite extensively described in the
PDP-11 manuals and will not be duplicated here. Several
examples of the syntax will be shown and by these means the
reader will be able to appreciate the methodology used on
the PDP-11.

1. Non-Overlaid Modules

At present, there are seven modules which are not
overlaid and, therefore, are the simplest to "build."
It is only necessary to compile these modules and taskbuild
(building an executable module and "linking" with external
references). To simplify the process even more, command
files are normally used for the process and were used for
building GIFTS.

For example, the module PRINT is built using the
following Command File:

```
PRINT/-P/-CP=PRINT,GIFTLI/LB
/
ACTFIL=15
MAXBUF =900
UNITS=15
ASG=TI:6
ASG=SY:7:8:9:10:11:12:13
ASG=SY:14:15
//
```

The switches used in the main line of the Command File are
necessary and more or less "boiler plate" switches. They

are explained in detail in the PDP-11 manuals. The
expression "GIFLIB/ " in the file indicates to the
taskbuilder that besides the PDP-11 system library, (on the
PDP-11, this is SYSLIB.OLB and need not be referred to in
the Command File as it is automatically called by TKB), a
library called GIFLIB.OLB is called in order to resolve
any external references. GIFLIB.OLB is actually made up
of the compiled object modules from the following seven
GIMS tape files:

```
LIBA1.
LIBA2.
LIBA3.
LIB 4.
LIBA5.FLP
DLIB.
LIBRAM.
```

The other lines in the command file for GIMS do warrant
some explanation as they are controlled by the GIMS system
size and parameters within the program. It is quite possible
that the parameters in the existing command files would
change in the future as the GIMS system is updated/modified.

        The term "MAXBUF" designates the size of the I/
buffer. The recordsize for several of the GIMS files is
quite large (up to 1634 bytes) but since not all files are
called by every modules, MAXBUF will vary between modules.
The size of MAXBUF for a particular module can be computed
by referring to the table on page M-N.2 of the GIMS
Systems Manual.

"UNITS" defines the maximum number of the logical units whereas "ACTFIL" assigns the number of active files which can be concurrently open. The latter is variable between modules and is quite important since the ACTFIL parameter causes allocation of memory at the time of task-building. As many of the modules are quite tightly "packed" into the 32K word allowable memory segment, the extra bytes available by adjusting this parameter become very important.

"ASG" fulfills the taskbuilder requirement that each logical unit have assigned to it a physical unit. Thus, in the PRINT command file, logical unit six is assigned (ASG) to the terminal (TI:) and all LU's between seven and 15, inclusively, are assigned arbitrarily to the "system" disc (SY:).

Without the command file, each of the steps would have to be individually typed in. Since a command file was built in this case, it is merely necessary to type:

> tkb print

It is worth noting here that if several modules are to be built, the command files may be imbedded in another command file. For example, take the command file BUILT.CMD which is made up of:

> tkb print
> tkb savex·
> tkb residu

This file can be executed by typing: @group.

2. Overlaid Modules

The majority of the GIFTS modules are overlaid.
Some of the overlay schemes are fairly complicated and are
difficult, due to the taskbuilder syntax, to enter at a
terminal. Therefore, as with the non-overlaid modules,
command files are used. However, now "indirect" or "Overlay
Description" files using "Overlay Description Language" (ODL)
are also used. (These files are commonly referred to as
"ODL files.")

The ODL file is built with the text editor for each
module and describes the overlay scheme for the module.
The file is then referred to by the TKB command file. For
example, the following is the command file used to build
the module BULKM in GIFTS. Notice on the right hand side
of the equal sign is the expression: BULKM/MP. The /MP
switch indicates to TKB that there exists a file on disc
called ".BULKM.ODL" which describes the overlay scheme for
the module (Figure 5). Note that no object modules are
referred to directly in this command file:

```
BULKM/MP/CP,,BULKM=BULKM/MP
ACTFIL=13
MAXBUF=960
UNITS=15
TOP=UT:5
ASG=SYD:7:8:9
ASG=TTO:10:11:12:13:14:15
//
```

The first line in the ODL file indicates the overlay scheme
in symbolic terms (i.e., A, B, C, etc.). The other lines

indicate the choice of object modules for the "Root" and the various overlays. There are syntax and command rules which obviously must be followed in building an ODL file. Such information is found in: "RSX-11M Task Builder Reference Manual." It is not the purpose here to elaborate on this syntax.

The Command and ODL files for building GIFTS exist for all overlaid modules and are on magnetic tape for the eventuality that the system will need to be rebuilt. These files are the core of the work necessary for building GIFTS. Anyone interested in vastly revising GIFTS would need to know the existing structure of GIFTS and then attempt to reconstruct the effect of the revisions on the size of modules. As stated above, some of the modules are very tightly packed, some taking up to greater than 99 percent of the available 32K words.

5. BUILDING OF LIBRARIES

There are two basic types of libraries built from the GIFTS files. The first type includes the two separate system libraries. The reason for having a second "system" library is that two GIFTS modules, BULKIB and RESULT, simply cannot fit into 32K words as normally built. Thus, a second nearly duplicate library is built using the "/EA:NONE" switch when compiling. The effect of this switch is to reduce the size of the object module by about ten percent.

The absence of the "trace" capability means that should
an error occur during program run time, the system will
not inform the user in which object module the error
occurred. Again, this "problem" occurs only in FULLIB
and RESULT.

The other type of library is called a "module" library.
That is, for every executable module where overlaying is
being used, a library of the object modules derived from
the individual program listing (vice the GIFTS system
library listings) is built. This approach allows the
analyst to "keep track" of which object modules are needed
for each overlaid module. Thus, this is a matter of con-
venience.

In Figure 5 are examples of how the two library types
are used. Note that in every case where the switch "/LB"
is seen, the preceding filename is the name of an object
library. Where the switch "/LB" is used alone, as in:
 IFTLIB/LB, the meaning is that a check through the library
GIFTID.OLB will be made to resolve references. Where a
colon is attached (i.e. "/LB:"), the MID system will expect
to find one or more specifically named object modules which
are to be designated as being part of a particular segment.

E. OVERLAY SCHEMES USED

The magnetic tape received from IBIM, University of
Arizona, included the overlay schemes used at the Naval

Postgraduate School for the building of GIFTS. The schemes
are actually in ODL file form. Changes to the overlay
scheme(s) would be completed by making revisions to the
respective ODL file and then rebuilding the respective
module(s).

Installing the GIFTS system on another computer system
could necessitate a revision to the schemes but the ODL
files are a good point for departure.

F. DELETION OF UNNECESSARY FILES

Along the path of building GIFTS, one accumulates
several files that are extraneous to the actual running of
the GIFTS system. If file deletions are not completed, an
accumulation of about 16,000 blocks of intelligence on
disc (about twenty percent of the maximum capacity of the
CDC 9762 disc drive) would be taken up by GIFTS. Since
the executable module files accumulate to only about 4000
blocks, file deletions (using PIP) should be completed.

The method for doing this on the PDP-11 can be found
in the appropriate PDP-11 Manual. Generally, it takes
the form:

PIP <u>Filename</u>.<u>Extension</u>;<u>Version</u>/DE

"Wild cards" are permitted for filenames, extensions and
version names/numbers. The version number (or wildcard)
must be included.

G. REBUILDING GIFTS

A.. files necessary to rebuild GIFTS exist on two
magnetic tapes. A listing of the contents of the respective
tapes are included as Appendix F. To rebuild GIFTS, it is
merely necessary to load the tapes and type the following
two commands:

```
FLX /RS=MT1:[*,*]BUILDT.CMD/DO
@BUILDT
```

The resulting process takes approximately six hours to
complete. A listing of BUILDT.CMD is included as
Appendix G.

V. PROCEDURE FOR REVISING GIFTS

A. MAKING MINOR CHANGES

It should be remembered that each module is listed
separately. In addition, there are five files of sub-
routine listings plus two assembly language files which
are included as part of the GIFTS system libraries (two).
It should be quite obvious that if a revision to a single
module listing is necessary then only that module will need
to be rebuilt.

On the other hand, if one library subroutine is changed
it would be wise to rebuild the entire system (unless the
modules containing the revised subroutine can be isolated).

1. Changing the System Library

The following steps should be completed in revising
the system libraries:

a. Edit (EDT) the listing (either LIBR1.NEW,
LIBR2.NEW, LIBR3.NEW, LIBR4.NEW or LIBR5.PEP);

b. Extract the subroutine(s) which have been
revised (in order that the entire library need not be
rebuilt);

c. Compile the subroutine twice-once with the
/CO:20 switch alone and again with the /TR:NONE switch;

d. Insert the object modules into the two
libraries - GIFTLIB and GLIB2 by using the LBR utility;

e. Rebuild the GIFTS system.

The last step is not quite as difficult as it
seems since the command and ODL files are already built
for this purpose. The entire rebuilding process can be
done by a series of TKB "@" statements. Such a command file,
called GIFTS5.CMD, is shown in Figure 6 and is included on
the tapes mentioned in section IV.G. By merely typing
@GIFTS5, the entire GIFTS system will be built in approxi-
mately one hour. The file depends, of course, on the
existence of the command files, ODL files, GIFTS system
libraries, and the respective module libraries to execute.
A listing of the files needed to execute GIFTS5.CMD are
shown in Figure 7.

2. Changing a Module Library

If only a single module listing is revised, then
it should not be necessary to build the entire GIFTS system.
In other words, the use of BUILDT.CMD is unnecessary here.
Instead, it would only be necessary to execute the steps
which are demonstrated in Appendix D. The OPTIM module
is used by way of example in Appendix D, but any overlaid
module would be "rebuilt" in the same manner.

43

For non-overlaid modules, it is necessary only to compile[10] and taskbuild using the provided command files.

E. MAJOR CHANGES

If a substantial number of changes to the GIFTS system were to be made, it may be necessary to rebuild the entire set of executable modules. Assuming that the command files are not to be revised,[11] the following steps would be followed:

1. Revise the respective listing(s) using the Text Editor (EDT);

2. Revise the two tapes using FLX;

3. Execute @BUILDT.

It should be obvious that if the two existing tapes are to be modified that a new set of tapes will need to be built. The FLX utility is the handler for this process. It should be noted that the present command file, BUILDT.XXX, is based on the existence of two separate tapes with the contents being as listed in Appendix F. In this appendix,

---

[10]It should be remembered that the default extension for a FORTRAN file on the PDP-11 is "FTN." The FORTRAN listings provided by IGEL had the extension "NEW." Therefore, when compiling these programs using the F4P compiler, use syntax as follows (for the file called REDOS.NEW): F4P REDOS=REDOS.NEW.

[11]If new subroutine(s) were added to an individual module, then the respective "ODL File" would also need to be revised. It is also possible that changes to existing subroutines could make the individual module greater than 32K with existing overlay schemes. Then, a revised scheme may be necessary and the ODL File would have to be revised.

it should also be noted that the UIC for the tape file is:

[20,1]. This UIC is presumed when BUILDT is executed.

C. UPDATING OF HELP FILE

There exists a file called GIFTS5.INF which contains
the information or data used by the "HELP" command from the
various GIFTS modules. It will be necessary to use an
editor to change this file. Revisions would be needed to
this file only if updates were received from the University
of Arizona.

VI. RECOMMENDATIONS

Several possibilities exist at the Naval Postgraduate
School for the enhancement of the GIFTS system. A TEKTRONIX
4081 computer is already present within the Mechanical
Engineering Department and could be used as an intelligent
terminal. That is, it would be possible to operate with
some of the GIFTS modules on a host computer such as the
PDP-11 with the smaller modules being used independently
on the 4081.

Of course, when the new mainframe replaces the currently
used IBM 360/67 in FY 1981, a worthwhile project would be
to install GIFTS on it.

In addition, it is recommended that the interface pro-
gram for the SAP4 system, which is currently available at
NPS, be obtained from IGEL, University of Arizona, in order
that the SAP4 and GIFTS can be "tied together."

```
BULKM VER. 5.02

TYPE JOB NAME
PLATE
JOB PLATE    BEING CREATED
* MSTEEL
> 1

* ETH,1
> 1
? 0.1
* KPOINT
> 1
? 2,,
> 2
? ,2,
> 3
? ,6,
> 4
? 6,6,
> 5
? 6,,
> 6
? 1.414,1.414,
> SLINE
> L23
? 2,3,5
> L15
? 1,5,5
> L34
? 3,4,4
> L45
? 5,4,4
> *
```

```
* CARC
> C12
? 1,6,2,7
>
* COMPLINE
> L35
? L34,L45
>
* GETY
> QM4
? 1,1
* GRID4
> QUARTER
? C12,L23,L35,L15
>
* KN
* LNAM
* GNAM
* PLOT
```

Figure 1 - Program Interaction for PLATE
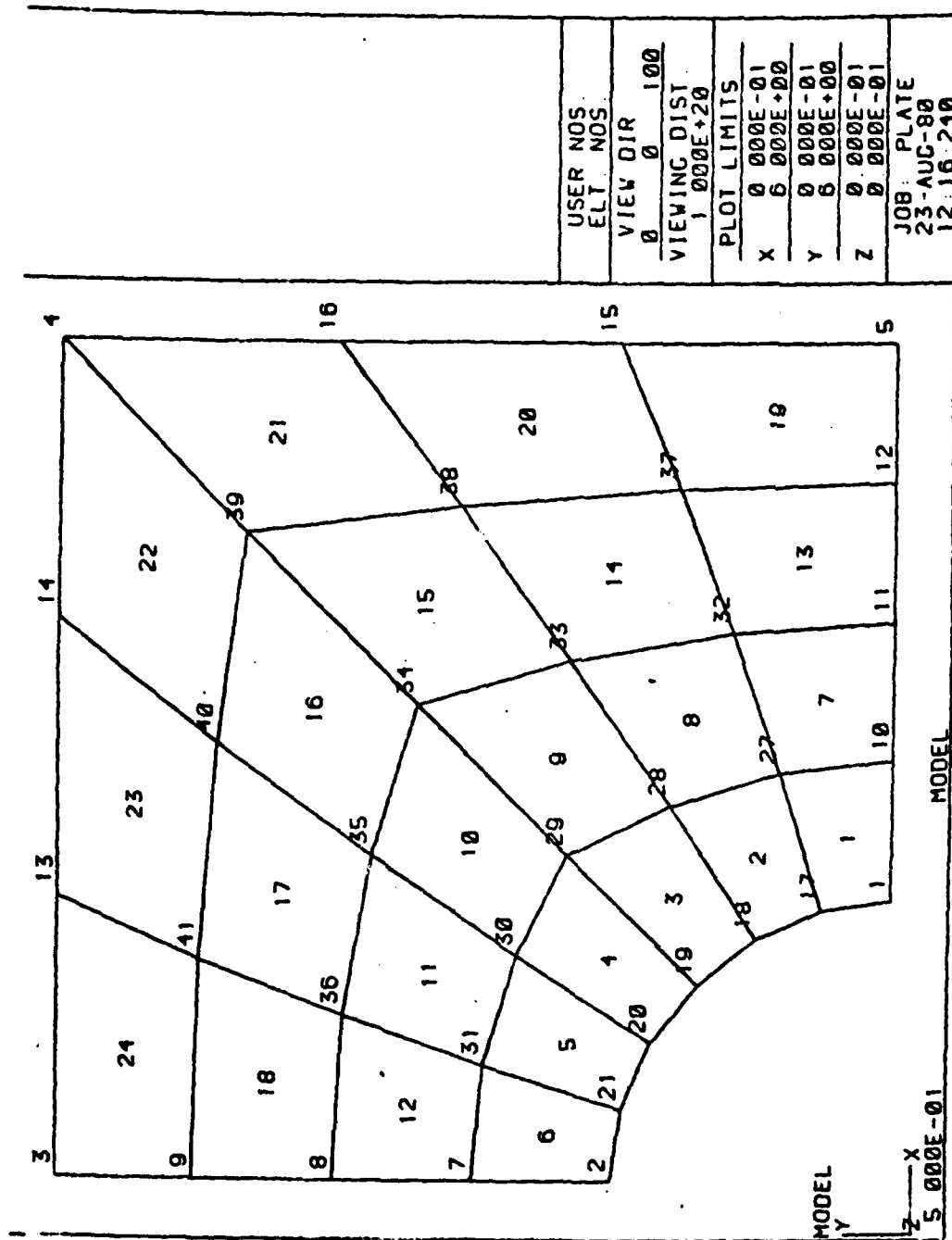
Figure 2 - GIFTS Presentation of PLATE Input
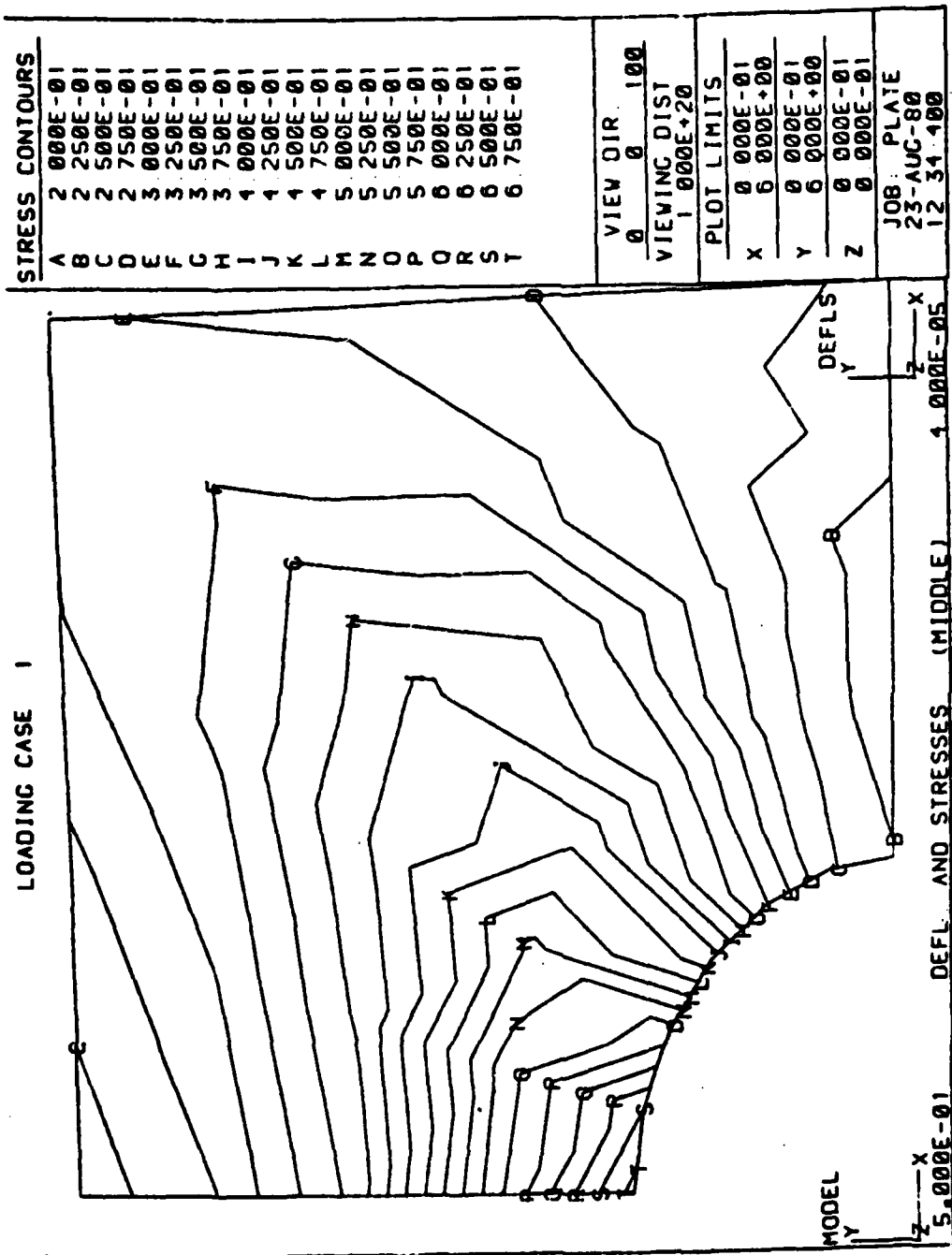
Figure 3 - CIFTS Presentation of Stress Contours JOB PLATE

49

```
>PIP PLATE */LI

DIRECTORY DP3:[20,1]
23-AUG-80 13:17

PLATE FIL;1     1.   23-AUG-80 12 07
PLATE MAT;1     1    23-AUG-80 12 07
PLATE THS;1     1    23-AUG-80 12 07
PLATE PTS;1     9    23-AUG-80 12 19
PLATE ELT;1    18.   23-AUG-80 12 07
PLATE LIN;1    10    23-AJC-80 12 07
PLATE CRD;1     7.   23-AUG-80 12 07
PLATE PAR;1     1.   23-AUG-80 12 15
PLATE LDS;1     7.   23-AUG-80 12 20
PLATE ELO;1    14.   23-AUG-80 12 20
PLATE SDY;1     1.   23-AUG-80 12 27
PLATE STF;1    23.   23-AUG-80 12 28
PLATE LDI;1     6.   23-AUG-80 12 30
PLATE CNI;1     6    23-AUG-80 12 30
PLATE DNS;1     5    23-AUG-80 12 30
PLATE STR;1     7    23-AUG-80 12 31

TOTAL OF 117 /160. BLOCKS IN 16  FILES
```

Figure 4 - Listing of Files Created by GIFTS JOB PLATE

50

```
MAIN      ROOT MAIN-(A,B,C1-C2,D,E,F,G,H,I,J,K,L,M)
MLIB      FCTR BULKM/LB MAIN DELINE GENLC LOCL-MLIB-GLB
A         FCTR CIFTLIB/LB LINPN DEFIN TWAIT DFIL
B         FCTR *BULKM/LB INITBM STOPBM-CIFTLIB/LB
          FCTR *BULKM/LB GENLNS CARC CMPLIN GENLE PARAM2 PARAM3 RSTCDS SL
INE-GLB
C1        FCTR *BULKM/LB GENCDS CENSE3 CENSE4 GLOB CRDCO3 CRDCO4 CRIDT GR
ID3
C2        FCTR *BULKM/LB CRID4 INCPT LOCC MULC OUTGPT PCRID-CIFTLIB/LB
D         FCTR *BULKM/LB SCALE PLOTBM LBLPLG-CIFTLIB/LB
E         FCTR *BULKM/LB COMBM HELP SWITCH-CIFTLIB/LB
F         FCTR *BULKM/LB ARBLIN LINI MERGLN THSS LGELTY MATL STANDM BMPLP
R-GLB
C         FCTR *BULKM/LB ACDPTR DCDPTR CRIDTI CRID3I CRID4I-CIFTLIB/LB
H         FCTR *BULKM/LB COMPOS CPTIME DCMPL DELCAL DCDPTR DLETBM CLASSB-
CIFTLIB/LB
I         FCTR *BULKM/LB ERRBM INFBM-CIFTLIB/LB
J         FCTR *BULKM/LB BEMSEC-CIFTLIB/LB
K         FCTR *BULKM/LB GENKPT KPOINT KCARC K2PARM K3PARM KSLINE-CIFTLIB
/LB
L         FCTR *BULKM/LB FNCTN PROJCT PRJECT-CIFTLIB/LB
M         FCTR *BULKM/LB POLYG-GLB-*(M1,M2)
M1        FCTR *BULKM/LB INPOLY-CIFTLIB/LB
M2        FCTR *BULKM/LB CALPOL-CIFTLIB/LB
CLB       FCTR CIFTLIB/LB
END
```

Figure 5 - Listing of BULKM ODL

51

BRO T1: BUILDING CIFTS
TKB @STRESS
TIME
TKB @EDITLB
TIME
TKB @SAVEK
TIME
TKB @RESIDU
TIME
TKB @BULKLB
TIME
TKB @STIFF
TIME
TKB @BULKM
TIME
TKB @DEFL
TIME
TKB @RESULT
TIME
TKB @BULKF
TIME
TKB @DECOM
TIME
TKB @EDITM
TIME
TKB @AUTOL
TIME
TKB @DEFCS
TIME
TKB @LOCAL
TIME
TKB @SUBS
TIME
TKB @TRAN1
TIME
TKB @TRANS
TIME
TKB @OPTIM
TIME
TKB @PRINT
TIME
TKB @TRAN2
TIME
TKB @REDCS
BRO T1: DONE BUILDING CIFTS

Figure 6 Listing of CIFTSS.CMD

```
CIFTLIB.OLB,1          STIFF.CMD,11
CLIB2.OLB,1            SUBS.CMD,11
OPTIM.OLB,1            TRAN1.CMD,11
DECOM.OLB,1           •TRANS.CMD,11
BULKL1.OLB,1           OPTIM.CMD,11
SUBS.OLB,1             STRESS.CMD,11
DEFCS.OLB,1            EDITLB.CMD,15
LOCAL.OLB,1            SAVEK.CMD,11
TRANS.OLB,1            PRINT.CMD,11
STRESS.OLB,1          RESIDU.CMD,11
DEFL.OLB,1            TRAN2.CMD,11
STIFF.OLB,1           REDCS.CMD,11
REDCS.OLB,1           CIFTSS.CMD,11
EDITLB.OLB,1          BULKM.CMD,26
EDITM.OLB,1           BULKLB.CMD,13
RESULT.OLB,1          DEFL.CMD,11
BULKM.OLB,1           RESULT.CMD,15
DECOM.ODL,53          BULKF.CMD,28
OPTIM.ODL,47          EDITM.CMD,12
BULKM.ODL,52          AUTOL.CMD,11
BULKLB.ODL,56         DECOM.CMD,14
DEFL.ODL,55           DEFCS.CMD,12
EDITM.ODL,52          LOCAL.CMD,11
LOCAL.ODL,65
SUBS.ODL,57
DEFCS.ODL,65
TRANS.ODL,61
STRESS.ODL,55
STIFF.ODL,52
REDCS.ODL,62
RESULT.ODL,64
EDITLB.ODL,56
```

Figure 7 - Listing of Files Needed to Execute CIFTSS.CMD

APPENDIX A

DESCRIPTION OF GIFTS MODULES[12]

MODEL GENERATION AND EDITING

BULKM

BULKM is an automated three dimensional plate and shell model generator. It is suitable for large continuous structure that can be easily modeled by repetitious generation of points and elements.

EDITM

EDITM is designed to update and correct BULKM models, although it can be used to generate simple models and ones too complex for BULKM.

DEFCS

DEFCS accepts information regarding external and dependent boundary nodes in a constrained substructure.

BULKS[13]

BULKS is a three dimensional solid model generator. One may ask for the display of the edges, and may add and display selected point and element slices.

---

[12] The descriptions here are taken from the "GIFTS User's Manual."

[13] BULKS, as of the date of this writing, is not yet implemented on the PDP-11. This is primarily due to its size.

LOAD AND BOUNDARY CONDITION
GENERATION, DISPLAY AND EDITING

BULKF

BULKF is intended to allow only those freedoms which a model can support, thereby relieving the user of the necessity of supressing all superfluous freedoms by hand.

BULKLB

BULKLB is a bulk load and boundary condition generator designed to apply load to models generated with BULKM. It may be used to apply distributed line and surface loads and masses, presribed displacements along lines and surfaces and inertial loads. Temperatures may also be applied to lines and surfaces.

EDITLB

EDITLB is a display and edit routine intended to provide local modification capability to loads and boundary conditions applied by BULKLB. It may also be used to generate simple loading on models, or loading on models not generated with BULKM. Temperatures may also be applied to elements. After DATB has been run, the thermal and combined loads may be examined.

LOADS[14]

LOADS is a load and boundary condition generator for solid models. Loads may be distributed on lines or

---

[14]LOADS, as of the date of this writing, is not yet implemented on the PDP-11. This is primarily due to size.

surfaces. Loads and boundary conditions may be displayed
on point slices.

GENERAL PURPOSE COMPUTATIONAL
AND RESULT DISPLAY MODULES

OPTIM

OPTIM is a band width optimization program. Although
GIFTS is designed to handle problems without size or band-
width restrictions, it is very important that the problem
be optimized before the solution procedes. Experience has
shown that run times can be reduced by a factor of two to
ten if the procedure is used. OPTIM may be called several
times in a row, until the best node numbering scheme has been
achieved.

STIFF

STIFF performs computation of the stiffness matrices and
assembles them into the master stiffness matrix.

DECOM

DECOM introduces kinematic boundary conditions, and
decomposes that stiffness matrix by the Cholesky method.

DEFL

DEFL computes the deflections from the current loading
conditions and the decomposed stiffness matrix. If tempera-
tures are present, thermal forces will be calculated and
added to the current applied loads before solution.

<u>STRESS</u>

STRESS computes the element stresses based on the current deflections.

<u>RESULT</u>

RESULT displays deflections and stresses. It has many options that may be used, at the discretion of the user, to transform the results for optimum comprehension.

<u>THE GIFTS NATURAL VIBRATION PACKAGE</u>

<u>AUTOL</u>

AUTOL is ordinarily used to generate starting loads for the subspace iteration to compute natural modes of vibration.

<u>SUBS</u>

SUBS performs a single subspace iteration to determine the model's natural modes. It must be repeated as many times as necessary to obtain convergence to the desired extent.

<u>THE GIFTS TRANSIENT RESPONSE PACKAGE</u>
<u>(DIRECT INTEGRATION)</u>

<u>TRAN1</u>

TRAN1 is to be run on a transient response model immediately after stiffness assembly. It is used to specify the time step to be used in the integration process.

56

TRAN2

TRAN2 is run after TRAN1 and DECOM. It computes the displacement matrix for time T.

TRANS

TRANS maintains and plots histograms of the displacements of up to four different freedoms.

GIFTS CONSTRAINED SUBSTRUCTURING PACKAGE

REDCS

Before a COSUB module may be used in a master analysis run, it must be preceded by program REDCS to form a reduced stiffness matrix and a reduced load matrix (if there are any loads associated with the COSUB).

APPENDIX E

LIST OF GIFTS MANUALS[15]

GIFTS USER'S REFERENCE MANUAL

"Contains complete and detailed description of all GIFTS commands and computational procedures. It is meant as a source of information for the experienced user."

GIFTS SYSTEMS MANUAL

"Contains detailed information on the code, data base and program structure. Useful for those undertaking program conversion or enhancement."

Though there is a great deal of detail concerning the UDB and program structure (for the ECLIPSE Computer), there is really insufficient information to get started in a "program conversion or enhancement." There are several terms and acronyms which are undefined in the description where knowledge of the other manuals are essential for understanding.

GIFTS PRIMER

". . . useful to new users, and to exercise the system on a new installation, or check out a new version of the program on an existing installation . . . Tutorial . . . Solved Examples."

This manual is excellent for the intended purposes. Anyone seriously intending to use the system should spend several hours with this manual and the computer.

---

[15]Remarks in quotation marks are taken directly from pages GPRIM-1-3 & 4 from the GIFTS Primer.

GIFTS INSTALLATION MANUAL

"Designed to help those attempting to install the program on their own system. Describes implementation and test procedures."

This manual, as of this writing, is not implemented.

It is hoped that with respect to the PDP-11, this thesis

provides some of the information needed.

GIFTS THEORETICAL MANUAL

"Contains mathematical fundamentals and algorithms underlying mesh generation, element characteristics and solution procedures. Of use to those wishing to assess the properties of the mathematical model used, or modify the program."

GIFTS MODELLING GUIDE

"Aimed at the program user. Discusses practical aspects of finite element modelling in general and pays particular attention to elements and procedures implemented in the GIFTS system."

Not implemented as of this writing.

GIFTS POCKET MANUAL

"A handy pocket-size reference manual containing complete, but terse, summary of information in the GIFTS Users Reference Manual. Used mainly as a quick reference manual to be used while working on a terminal by the experienced user."

Emphasize experienced!

APPENDIX C

LISTING OF PROGRAM: FILSOR

```
C**************FILE SORTER*******************     FIL00030
C*********WRITTEN BY JOHN T. SHELDON*********     FIL00040
C*******************************************     FIL00050
C*** LAST UPDATED 8/25/80 BY JTS                 FIL00060
C                                                FIL00070
C THIS PROGRAM IS A FILE SORTER. IT WAS SPECIFICALLY  FIL00080
C WRITTEN WITH THE "GIFTS" SYSTEM IN MIND. THE GIFTS  FIL00090
C SOURCE LISTINGS HAVE THE FOLLOWING CHARACTERISTICS: FIL00100
C                                                FIL00110
C      1)  THERE ARE NO BLOCK DATA SUBROUTINES;  FIL00120
C                                                FIL00130
C      2)  SOME OF THE CONCATENATED LISTINGS START  FIL00140
C          WITH A MAIN PROGRAM;                  FIL00150
C                                                FIL00160
C      3)  FIVE OF THE FORTRAN LISTINGS INCLUDE ONLY  FIL00170
C          SUBROUTINES OR FUNCTIONS(IE NO MAIN   FIL00180
C          PROGRAMS);                            FIL00190
C                                                FIL00200
C      4)  IN ALL CASES, THE PROGRAM, SUBROUTINES AND  FIL00210
C          FUNCTIONS(PSF) WILL HAVE TO BE COMPILED;  FIL00220
C                                                FIL00240
C      5)  IN SOME CASES, ENTIRE SYSTEMS OF SORTED  FIL00250
C          SUBROUTINES WILL HAVE TO BE COMPILED  FIL00260
C          WITH THE "/TR:NONE" SWITCH IN USE;    FIL00270
C                                                FIL00280
C      6)  IN ALL CASES, THE SORTED PSF'S WILL HAVE  FIL00290
C          TO BE PUT IN A LIBRARY.               FIL00300
C                                                FIL00310
C      7)  THE COMMENT STATEMENTS PRECEDING THE FIRST  FIL00320
C          EXECUTABLE STATEMENT ARE DISCARDED;   FIL00330
C                                                FIL00340
C      8)  THE INPUT FILE IS UNAFFECTED BY THIS PROGRAM;  FIL00350
C                                                FIL00360
C      9)  THE "END" STATEMENT BEGINS IN COLUMN 7  FIL00370
C                                                FIL00380
C                                                FIL00390
C                                                FIL00400
      BYTE YES,NO,ANS,ANANS(40),LINE(72),OBLANK  FIL00410
      BYTE F,E,NN,C,S,DD,SFLAG                    FIL00420
                                                 FIL00430
```

60

```
      LOGICAL OK,ENDIT                                                   FIL00440
      COMMON SFLAG                                                       FIL00450
      DATA F,E,NN,DD /1HF,1HE,1HN,1HD/                                   FIL00460
     1 IEND,OFLAG,SFLAG,OBLANK/0,0,0,1H /                                FIL00470
     2 YES,NO,C,S/1HY,1HN,1HC,1HS/                                       FIL00480
      ITRACE=0                                                           FIL00490
C                                                                        FIL00500
      WRITE(6,100)                                                       FIL00510
      WRITE(6,106)                                                       FIL00520
C  FIND OUT IF /TR:NONE SWITCH IS DESIRED                                FIL00530
      READ(5,101) ANANS(1)                                              FIL00540
      IF(ANANS(1).EQ.YES)ITRACE=1                                        FIL00550
      IF(ITRACE.EQ.1)SFLAG=1                                             FIL00560
C  GET NAME OF FILE TO BE SORTED                                         FIL00570
      WRITE(5,104)                                                       FIL00580
      READ(5,101)ANANS                                                   FIL00590
C  OPEN STATEMENTS REQUIRES THAT LAST BYTE IN NAME BE 0                  FIL00600
      ANANS(40)=0                                                        FIL00610
C  OPEN INPUT FILE                                                       FIL00620
      OPEN(UNIT=1,NAME=ANANS,TYPE='OLD',ACCESS='SEQUENTIAL',             FIL00630
     1 DISP='SAVE')                                                      FIL00640
C  LOOKING FOR FIRST EXECUTABLE STMNT IN PROGRAM/SUBROUTINE OR           FIL00650
      FUNCTION(PSF)                                                      FIL00660
   15 CALL RDLINE(LINE)                                                  FIL00670
C  "N" EQUALS THE NUMBER OF LAST NON-BLANK CHARACTER                     FIL00680
C  IN THE LINE                                                           FIL00690
      N=0                                                                FIL00700
      DO 17 I=1,72                                                       FIL00710
   17 IF(LINE(I).NE.OBLANK)N=I                                           FIL00720
C  IF LINE IS A COMMENT CARD, IGNORE IT                                  FIL00730
      IF(LINE(1).EQ.C)GO TO 15                                           FIL00740
C  BLANK LINE?                                                           FIL00750
      IF(N.EQ.0)GO TO 15                                                 FIL00760
C  BEGINNING OF PSF(HOPEFULLY)                                           FIL00770
C  GET NAME OF FILE                                                      FIL00780
      CALL NAMEFL(LINE,ANANS)                                            FIL00790
C  PREPARE AND STORE COMMAND FILE INPUTS                                 FIL00800
      CALL CMDFIL(ANANS)                                                 FIL00810
      ANANS(40)=0                                                        FIL00820
C  OPEN OUTFILE                                                          FIL00830
      OPEN(UNIT=2,NAME=ANANS,TYPE='NEW',ACCESS='SEQUENTIAL',             FIL00840
     1 DISP='SAVE')                                                      FIL00850
C  START STORING AND READING UNTIL END STATEMENT ENCOUNTERED             FIL00860
      ENDIT=.FALSE.                                                      FIL00870
   20 WRITE(2,101)(LINE(I),I=1,N)                                        FIL00890
```

61

```
      IF(ENDIT.EQ..TRUE.)GO TO 25
C     LOOK FOR "E" IN COLUMN 7 AND "N" IN COLUMN 8
      ENDIT=(LINE(7).EQ.E).AND.(LINE(8).EQ.N)
C     MAKE OTHER CHECKS TO ENSURE THIS IS AN END STATEMENT
      IF(.NOT.(ENDIT.AND.(LINE(9).EQ.DD.AND.LINE(10).EQ.32)))
     1ENDIT=.FALSE.
      IF(LINE(1).EQ.C)ENDIT=.FALSE.
      N=0
      DO 21 I=1,72
   21 IF(LINE(I).NE.OBLANK)N=I
      GO TO 20
C
C     DONE WITH THIS OUTFILE
   25 CLOSE(UNIT=2,DISP='SAVE')
      GO TO 15
  100 FORMAT(20X,' ***FILE SORTER***',/)
  101 FORMAT(80A1)
  104 FORMAT('$TYPE IN NAME OF FILE INCLUDING EXTENSION:')
  105 FORMAT(1X,80A1)
  106 FORMAT('$'//TR:NONE" SWITCH DESIRED FOR COMPILE?(Y OR N):')
      END
      SUBROUTINE CMDFIL(FILNAM)
C*************************************************************
C****************WRITTEN BY JOHN T. SHELDON*****************
C*************************************************************
C*** LAST UPDATED ON 8/25/80 BY JTS
C*************************************************************
C     THIS SUBROUTINE IS PART OF THE FILSOR PACKAGE
C     IT TAKES THE FILNAME(FILNAM) AND DETERMINES THE
C     LIB.CMD AND STUFF.CMD INPUTS.
C
C     LIB.CMD IS A FILE OF STATEMENTS WHICH ARE
C     USED ALONG WITH THE F4P COMPILER ON THE
C     PDP11.  THE OUTPUT LINES ARE OF THE FORM:
C
C /TR:NONE'
C
C     THE /CO:20 SWITCH HAS BEEN INSERTED AS
C     A FEW OF THE SUBROUTINES WILL HAVE MORE
C     THAN THE DEFAULT CONTINUATION LINES.
C
C     THE /TR:NONE SWITCH IS AN OPTION WHICH
C     IS USED IF SPACE IS AT A PREMIUM(AS IN
C     THE RESULT AND BULKLB MODULES OF GIFTS.
```

```
FIL00920
FIL00930
FIL00940
FIL00950
FIL00960
FIL00970
FIL00980
FIL00990
FIL01000
FIL01010
FIL01020
FIL01030
FIL01040
FIL01050
FIL01060
FIL01070
FIL01080
FIL01090
FIL01100
FIL01110
FIL01120
FIL01130
FIL01140
FIL01150
FIL01160
FIL01170
FIL01180
FIL01190
FIL01200
FIL01210
FIL01220
FIL01230
FIL01240
FIL01250
FIL01260
FIL01270
FIL01280
FIL01290
FIL01300
FIL01310
FIL01320
FIL01330
FIL01340
FIL01350
FIL01360
FIL01370
FIL01380
FIL01390
```

62

```
C     STUFF.CMD IS A FILE OF INDIVIDUAL "LBR"
C     COMMANDS WHICH "STUFF" THE COMPILED
C     OBJECT MODULES IN A LIBRARY CALLED
C     (ARBITRARILY): L1.OLB
C
C     THE ABOVE TWO FILES ARE EXECUTED IN THE FOLLOWING
C     ORDER AND WITH THE SAME SYNTAX:
C
C       @F4PLIB
C       @STUFF
C
C     ERRORS DURING COMPILE WOULD BE PRINTED. OTHERWISE
C     NO OUTPUT SHOULD BE EXPECTED FROM THE FIRST LINE.
C     STUFF, ON THE OTHERHAND, WILL PRINT EACH COMMAND
C     LINE.
C
      BYTE ISLASH,IEQ,SFLAG
      BYTE FILNAM(40),LIBNAM(2),IDOT,LOUTPT(40),MOUTPT(40)
      DIMENSION LDUP(20),MDUP(5),LLDUP(7)
      COMMON SFLAG
      EQUIVALENCE (LOUTPT(1),LDUP(1)),(MOUTPT(1),MDUP(1))
      DATA IEQ/1H=/
      DATA LDUP(1),LDUP(2),MDUP(1),MDUP(2)/2H  ,2H  ,2HLB,2HR  /
     1 IFLAG,LIBNAM(1),LIBNAM(2),OBLANK/O,1HL,1H1,1H /
     2 IDOT,MDUP(3),MDUP(4),MDUP(5)  /1H.,2HLI,2H/I,2HN=/
     3 LLDUP(1),LLDUP(2),LLDUP(3)/2H/C,2HO:,2H2O/
     4 LLDUP(4),LLDUP(5),LLDUP(6)/2H/T,2HR:,2HNO/
     5 LLDUP(7)/2HNE/
C     SFLAG=1 INDICATES /TR:NONE SWITCH IS TO BE LEFT IN
C     SFLAG =0 WILL CAUSE THE ARRAY HOLDING THIS EXPRESSION
C                   TO BE ZEROED
      IF(SFLAG.EQ.1)GO TO 5
      DO 3 I=4,7
    3 LLDUP(I)=0
      SFLAG=0
C     CHECK TO SEE IF FIRST PASS THROUGH SUBROUTINE
C     IF IT IS, MUST OPEN FILES
    5 IF(IFLAG.EQ.1)GO TO 10
C     OPEN COMMAND FILES
      LU=3                       "STUFF.CMD" FILE
      OPEN(UNIT=3,NAME='STUFF.CMD',TYPE='UNKNOWN',
     1 ACCESS='APPEND',DISP='SAVE')
```

```
FILO1400
FILO1410
FILO1420
FILO1430
FILO1440
FILO1450
FILO1460
FILO1470
FILO1480
FILO1490
FILO1500
FILO1510
FILO1520
FILO1530
FILO1540
FILO1550
FILO1560
FILO1570
FILO1580
FILO1590
FILO1600
FILO1610
FILO1620
FILO1630
FILO1640
FILO1650
FILO1660
FILO1670
FILO1680
FILO1690
FILO1700
FILO1710
FILO1720
FILO1730
FILO1740
FILO1750
FILO1760
FILO1770
FILO1780
FILO1790
FILO1800
FILO1810
FILO1820
FILO1830
FILO1840
FILO1850
FILO1860
FILO1870
```

```
C     LU=4    "LIB.CMD" FILE                                    FILO1880
      OPEN(UNIT=4,NAME='LIB.CMD ',TYPE='UNKNOWN',              FILO1890
     1  ACCESS='APPEND',DISP='SAVE')                           FILO1900
      IFLAG=1                                                  FILO1910
10    DO 15  I=1,16                                            FILO1920
      IF(FILNAM(I).EQ.OBLANK.OR.FILNAM(I).EQ.IDOT)GO TO 16     FILO1930
C     N IS THE NUMBER OF CHARACTERS IN THE NAME OF THE PSF     FILO1940
      N=I                                                      FILO1950
15    CONTINUE                                                 FILO1960
C                                                              FILO1970
C     M,N---                                                   FILO1980
C OF CHARACTERS IN "FILNAM"                                    FILO1990
16    M=N                                                      FILO2000
C                                                              FILO2010
C     BUILD LIB.CMD FILE INPUT                                 FILO2020
      N=N+4                                                    FILO2030
20    DO 20  I=5,N                                             FILO2040
      LOUTPT(I)=FILNAM(I-4)                                    FILO2050
      N=N+1                                                    FILO2060
      LOUTPT(N)=IEQ                                            FILO2070
30    DO 30  I=N+1,N+4                                         FILO2080
      LOUTPT(I)=FILNAM(I-M-5)                                  FILO2090
      DO 35  I=N+M+1,26                                        FILO2100
35    LOUTPT(I)=OBLANK                                         FILO2110
      DO 36  I=14,20                                           FILO2120
36    LDUP(I)=LLDUP(I-13)                                      FILO2130
      WRITE(4,100)LOJTPT                                       FILO2140
C     BUILD STJFF.CMD FILE INPUT                               FILO2150
40    DO 40  I=11,M+10                                         FILO2160
      MOUTPT(I)=FILNAM(I-10)                                   FILO2170
      DO 45  I=M+11,49                                         FILO2180
45    MOUTPT(I)=OBLANK                                         FILO2190
      WRITE(3,100)MOUTPT                                       FILO2200
      RETURN                                                   FILO2210
100   FORMAT(80A1)                                             FILO2220
      END                                                      FILO2230
      SUBROUTINE NAMEFL(LINE,ANANS)                            FILO2240
C***********************************************************   FILO2250
C*****************WRITTEN BY JOHN T. SHELDON**************** FILO2260
C***********************************************************   FILO2270
C*** LAST UPDATED ON 8/25/80 BY JTS                          FILO2280
C***********************************************************   FILO2290
C                                                              FILO2300
C     THIS SUBROUTINE IS A PART OF THE FILSOR PACKAGE.         FILO2310
C                                                              FILO2320
C     ITS PURPOSE IS TO OBTAIN THE NAME OF THE "OUTFILE"       FILO2330
```

64

```
C     BE IT A MAIN PROGRAM, SUBROUTINE OR FUNCTION.                    FIL02360
C                                                                      FIL02370
C     A LINE OF DATA IS ENTERED(LINE) AND THE PSF IS                   FIL02380
C     RETURNED(ANANS)                                                  FIL02390
      BYTE LINE(72),ANANS(40),S(12),F(8),OBLANK,PAREN                  FIL02400
      BYTE IDOT,FF,TT,NN,SFLAG                                         FIL02410
      COMMON SFLAG                                                     FIL02420
      DATA S(1),S(2),S(3),S(4),S(5)/1HS,1HU,1HB,1HR,1HO/               FIL02430
     1     S(6),S(7),S(8),S(9),S(10)/1HU,1HT,1HI,1HN,1HE/              FIL02440
     2     F(1),F(2),F(3),F(4),F(5)/1HF,1HU,1HN,1HC,1HT/               FIL02450
     3     F(6),F(7),F(8),OBLANK,PAREN/1HI,1HO,1HN,1H ,1H(/            FIL02460
     4     FF,TT,NN,IDOT,OFLAG    /1HF,1HT,1HN,1H.,0/                  FIL02470
C     ZERO ANANS                                                       FIL02480
      IF(OFLAG.EQ.0)GO TO 11                                           FIL02490
      DO 10 I=1,40                                                     FIL02500
10    ANANS(I)=OBLANK                                                  FIL02510
11    CONTINJE                                                         FIL02520
      OFLAG=1                                                          FIL02530
C     LOOK FOR "SUBROUTINE" OR "FUNCTION"                              FIL02540
      N=I                                                              FIL02550
      DO 15 I=7,50                                                     FIL02560
C     LOOKING FOR LETTER "S"                                           FIL02570
      IF(LINE(I).EQ.S(1))GO TO 20                                      FIL02580
C     DIDN'T FIND S. HOW ABOUT AN "F"                                  FIL02590
15    IF(LINE(I).EQ.F(1))GO TO 30                                      FIL02600
C     THIS MUST BE A MAIN PROGRAM SINCE THERE IS NO LETTER             FIL02610
C     "S" OR "F" IN THE FIRST EXECUTABLE LINE.                         FIL02620
16    CONTINJE                                                         FIL02630
C     THIS MEANS THAT "ANANS" CONTAINS                                 FIL02640
C     (RIGHT FILE NAME BUT WRONG EXTENSION                             FIL02650
C     (THIS IS TRUE SINCE THE MAIN PROGRAM IN THE GIFTS                FIL02660
C     LISTINGS ARE ALWAYS THE FIRST IN THE PACKAGE.                    FIL02670
C     THEREFORE, TO HAVE GOTTEN TO THIS POINT WE'RE                    FIL02680
C     LOOKING FOR A MAIN PROGRAM. ANANS HASN'T BEEN                    FIL02690
C     CHANGED SINCE IT WAS USED TO GET THE NAME OF                     FIL02700
C     THE INPUT FILE.                                                  FIL02710
      DO 17 I=1,6                                                      FIL02720
      II=I                                                             FIL02730
      IF(ANANS(I).EQ.IDOT)GO TO 18                                     FIL02740
17    CONTINUE                                                         FIL02750
      II=II+1                                                          FIL02760
18    ANANS(II+1)=FF                                                   FIL02770
      ANANS(II+2)=TT                                                   FIL02780
      ANANS(II+3)=NN                                                   FIL02790
      ANANS(40)=0                                                      FIL02800
      RETURN                                                           FIL02810
C     CHECK IF A "SUBROUTINE".  IF NOT THEN MUST                       FIL02820
                                                                       FIL02830
```

65

```
C      BE THE BEGINNING OF THE MAIN PROGRAM                      FILO2840
  20   IDIF=N-1                                                  FILO2850
       DO 21 I=N,N+9                                             FILO2860
  21   IF(LINE(I).NE.S(I-IDIF))GO TO 16                          FILO2870
C      MUST BE A SUBROUTINE                                      FILO2880
C      FIRST ZERO ANANS                                         FFILO2890
       DO 22 I=1,40                                             FFILO2900
  22   ANANS(I)=0                                                FILO2910
       N=N+10                                                    FILO2920
       DO 23 I=N,N+5                                             FILO2930
  23   IF(LINE(I).NE.DBLANK) GO TO 24                            FILO2940
  24   N=I                                                       FILO2950
       IDIF=N-1                                                  FILO2960
  25   IF(LINE(I).EQ.DBLANK.OR.LINE(I).EQ.PAREN)GO TO 26         FILO2970
       ANANS(I-IDIF)=IDOT                                       FFILO2980
  26   ANANS(I-IDIF)=LINE(I)                                     FILO2990
       ANANS(I-IDIF+1)=FF                                        FILO3000
       ANANS(I-IDIF+2)=TT                                        FILO3010
       ANANS(I-IDIF+3)=NN                                        FILO3020
       ANANS(40)=0                                               FILO3030
       RETURN                                                    FILO3040
C      CHECK IF A "FUNCTION". IF NOT, MUST THE BEGINNING OF A PROGRAM  FILO3050
                                                                 FILO3060
  30   IDIF=N-1                                                  FILO3070
       DO 31 I=N,N+7                                             FILO3080
  31   IF(LINE(I).NE.F(I-IDIF))GO TO 16                          FILO3090
C      MUST BE A FUNCTION                                        FILO3100
C      FIRST ZERO ANANS                                         FFILO3110
       DO 32 I=1,40                                              FILO3120
  32   ANANS(I)=0                                                FILO3130
       N=N+8                                                     FILO3140
       DO 33 I=N,N+5                                             FILO3150
  33   IF(LINE(I).NE.DBLANK)GO TO 34                             FILO3160
  34   N=I                                                       FILO3170
       IDIF=N-1                                                  FILO3180
       DO 35 I=N,N+6                                             FILO3190
  35   IF(LINE(I).EQ.DBLANK.OR.LINE(I).EQ.PAREN)GO TO 26         FILO3200
       ANANS(I-IDIF)=LINE(I)                                     FILO3210
       GO TO 26                                                  FILO3220
 102   FORMAT(80A1)                                              FILO3230
       END                                                       FILO3240
       SUBROUTINE RDLINE(LINE)                                   FILO3250
C**************************************************************  FILO3260
C*********************WRITTEN BY JOHN T. SHELDON**************** FILO3270
C***                                                            FILO3280
C *** LAST UPDATED ON 8/25/80 BY JTS                            FILO3290
                                                                 FILO3300
                                                                 FILO3310
```

66

```
C***********************************************
C     THIS SUBROUTINE IS PART OF THE FILSOR PACKAGE          FILO3320
C***********************************************             FILO3330
C                                                            FILO3340
C     THE PURPOSE OF THIS SUBROUTINE IS TO READ A            FILO3350
C     LINE FROM THE INPUT FILE.  IF AN "EOF" IS              FILO3360
C     REACHED, ALL OPEN FILES ARE CLOSED AND THE             FILO3370
C     FILSOR PROGRAM IS STOPPED HERE.                        FILO3380
C                                                            FILO3390
C                                                            FILO3400
C                                                            FILO3410
      BYTE LINE(72)                                          FILO3420
      READ(1,100,END=10)LINE                                 FILO3430
      RETURN                                                 FILO3440
   10 DO 20 I=1,4                                            FILO3450
   20 CLOSE(UNIT=I)                                          FILO3460
      STOP                                                   FILO3470
  100 FORMAT(80A1)                                           FILO3480
      END                                                    FILO3490
                                                             FILO3500
```

67

SAMPLE SESSION WITH FILSOR

The following is a listing from an actual run with
FILSOR. It has been annotated to indicate what actually
is going on and the reasons for the various steps.

The BUILDT.CMD file executes this entire process
"automatically" with the exception that the FILSR2 ver-
sion of FILSOR is needed as well as the STUFFL.TSK file
(which generates the answers to the FILSOR questions
asked below). BUILDT, of course, also reads the necessary
files from magnetic tape.

```
>PIP/LI


DIRECTORY DP3 [160,53]              (1)
30-AUG-80 15 46

OPTIM.ODL ,47        1               30-AUG-80 15 33
FILSOR.TSK ,2        51       C      30-AUG-80 15 33
OPTIM.NEW ,1         53.             30-AUG-80 15 33
OPTIM.CMD ,10        1.          .   30-AUG-80 15 33
GIFTLIB OLB ,1       592      C      30-AUG-80 15.45

TOTAL OF 698./698. BLOCKS IN 5. FILES



>RUN FILSOR                          (2)
                    ***FILE SORTER***

"/TR:NONE" SWITCH DESIRED FOR COMPILE?(Y OR N) N
TYPE IN NAME OF FILE INCLUDING EXTENSION OPTIM NEW
TT36  --  STOP

>F4P @LIB                            (3)
>PIP *.OBJ/LI                        (4)


DIRECTORY DP3:[160,53]
30-AUG-80 15:48

OPTIM OBJ ,1         1               30-AUG-80 15:46
BAND.OBJ ,1          6.              30-AUG-80 15 47
INOPT.OBJ ,1         5.              30-AUG-80 15:47
OPT.OBJ ,1           10.             30-AUG-80 15:47
SWAP OBJ ,1          3               30-AUG-80 15:47
TEROPT.OBJ ,1        14.             30-AUG-80 15:47

TOTAL OF 39./50. BLOCKS IN 6. FILES           (5)



>LBR L1/CR 39 .6. 6 .OBJ             (6)
>@STUFF                              (7)
>LBR L1/IN-OPTIM
>LBR L1/IN-BAND
>LBR L1/IN-INOPT
>LBR L1/IN-OPT
>LBR L1/IN-SWAP
>LBR L1/IN-TEROPT
>@ <EOF>
>PIP OPTIM.OLB-L1.OLB/RE             (8)
>TKB @OPTIM                          (9)
```

```
>PIP/LI                                (10)


DIRECTORY DP3 [160,53]
30-AUG-80 15 51

  STUFF.CMD ,1           1             30-AUG-80  15 46
  OPTIM.ODL ,47          1.            30-AUG-80  15 33
  LIB.CMD ,1            1.            30-AUG-80  15 46
  FILSOR.TSK ,2         51.      C    30-AUG-80  15 33
  OPTIM.NEW ,1         53            30-AUG-80  15 33
  OPTIM.FTN ,1          2.            30-AUG-80  15 46
  OPTIM.CMD ,10         1.            30-AUG-80  15 33
  BAND.FTN ,1           7.            30-AUG-80  15 46
  INOPT.FTN ,1          7.            30-AUG-80  15 46
  OPT.FTN ,1           18.            30-AUG-80  15 46
  SWAP.FTN ,1           4.            30-AUG-80  15 46
  TEROPT.FTN ,1        14.            30-AUG-90  15 46
  OPTIM.OBJ ,1          1.            30-AUG-80  15 46
  BAND.OBJ ,1           6.            30-AUG-80  15 47
  INOPT.OBJ ,1          5.            30-AUG-80  15 47
  OPT.OBJ ,1           10.            30-AUG-80  15 47
  SWAP OBJ ,1           3.            30-AUG-80  15 47
  TEROPT.OBJ ,1        14.            30-AUG-80  15 47
  GIFTLIB.OLB ,1       592.     C    30-AUG-80  15 45
  OPTIM.OLB ,1         40.            30-AUG-80  15 48
  OPTIM.TSK ,1        209.     C    30-AUG-80  15 49
  OPTIM.STB ,1          6.            30-AUG-80  15 51

TOTAL OF 1046 /1086 BLOCKS IN 22. FILES


>PIP *.FTN,*/DE                        (11)
>PIP *.OBJ,*/DE
>PIP * CMD,*/DE
>PIP * ODL ,*/DE
>PIP *.OLB ,*/DE
>PIP FILSOR.TSK ,*/DE
>PIP OPTIM NEW ,*/DE
>PIP/LI

                                       (12)

DIRECTORY DP3 [160,53]
30-AUG-80 15 52

  OPTIM TSK ,1        209.     C    30-AUG-80  15 49
  OPTIM STB ,1          6.            30-AUG-80  15 51

TOTAL OF 215 /219 BLOCKS IN 2. FILES



   >
```

(1) The first thing done is a "PIP/LI" which lists all files in the directory. In this case, the response indicates that we're in directory 160,53. The files listed are all the files needed to build OPTIM. If RESULT or BULMLIB were being built, then GLIB2 would replace OPTILIB as the GITIS "System" library. Also, as a means of differentiating the BULMIB and RESULT module libraries (see section IV.D) these libraries are arbitrarily referred to, in the command files, as BULML1 and RESUL1, respectively.

(2) FILSOR is executed. It responds by asking two questions before proceeding.

(3) The sorted program/subroutines are compiled separately using the command file: LIB:SMD (which was generated by FILSOR (see listing in item (10) below)). If an error were generated during compile, the compiler would indicate which subroutine had the error(s). This would not, however, inhibit the further completion of compilation.

(4) This is a listing of the "Object" files generated by the F4P @LIB command.

(5) Note that 39 blocks in six files were generated. These numbers are critical in that they are used to create a library in the next step.

(6) Using the "LBR" utility, a library "L1.OLB" is created. The decimal points are parts of the syntax in this command as the omission of them indicates octal numbers. The name

71

"L1" is used only because the "SNIFF.MD" file, built by MRSOR is looking, arbitrarily, for a object library called L1.

(7) Library L1.OLB is "stuffed." In the case of this command file, each command is subsequently listed until and ⟨EOF⟩ is encountered. In this example, six object modules have been inserted into library L1.OLB.

(8) Using PIP, the library L1.OLB is renamed: OPTIM.OLB. This is the library name which will be used by OPTIM.CMD when taskbuilding OPTIM.

(9) OPTIM is finally "taskbuilt."

(10) A listing of the files which have been generated while building the executable module, OPTIM.TSK, and the symbol table file, OPTIM.STB. Note that the sum of the space taken up by the files is over 1000 blocks.

(11) Housekeeping. Those files unnecessary to the execution of the OPTIM module are deleted by the seven PIP directives shown.

(12) A listing of what remains: the two files necessary to execute optimization.

APPENDIX F

LISTING OF BUILGT.CMD

```
TYPE
.ASK QUES ARE THE TWO TAPES MOUNTED
.IFF QUES .GOTO 200
.ASK QUES HAVE YOU DONE A "DIR/FR"
.IFF QUES .GOTO 200
.ASK QUES ARE THERE AT LEAST 9K BLOCKS AVAILABLE
.IFF QUES .GOTO 200
.ASK QUES THIS IS GOING TO TAKE ABOUT 6 HOURS. READY
.IFF QUES .GOTO 200

FLX /RS=MT1: 20,1 FILSR2.TSK/DO
FLX /RS=MT1: 20,1 STUFF.TSK/DO
PIP *.*=FILSR2.FSR/CO
PIP *.*=STUFF.TSK/CO
PIP *.TSK/DU
FLX /RS=MT1: 20,1 *.CMD/DO
FLX /RS=MT: 20,1 *.FSR.MEM/DO
FLX /RS=MT: 20,1 *.FSR.MEM/DO
FLX /RS=MT: 20,1 *.FSR.MEM/DO
FLX /RS=MT: 20,1 *.FSR.MEM/DO
FLX /RS=MT: 20,1 *.FSR.MEM/DO
FLX /RS=MT: 20,1 *.BUIL.MAC/DO
FLX /RS=MT: 20,1 *.FSR.MAC/DO
```

73

```
PIP STUFF
RUN FILSR2
RUN FILSR2
RUN FILSR2
RUN FILSR2
RUN FILSR2
LBR 11/CR:636.:384.:384.:??
FAP 3LIB
QSTUFF
PIP 4IFTLIB.OLB=11.OLB/RE
PIP *.FT*;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/RE
TUFF
RUN FILSR2
RUN FILSR2
RUN FILSR2
RUN FILSR2
PIP *.MED;*/DE
PIP *.PPO;*/RE
LBR 11/CR:500.:384.:384.:??
FAP OLIB
QSTUFF
PIP OLIB2.OLB=11.OLB/RE
MAC OFIL=PFIL
MAC IPFMAP=IPFMAP
PIP *.MAC;*/D
LBR ?IFTLIB/LK=OFIL,IPFMAP
LBR OLIB2/14=PFIL,IPFMAP
PIP *.FT*;*/RE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
```

74

```
PIP STUFF.CMD;*/DE
FLX /RS=MT: 29,1 STUFF.MEM/DO
TIME
RUN FILSR2
LBR LI/CR:290.:100.:100.:98.1
F4P @LIB
@STUFF
PIP STUFF.OLB=11.OLB/RF
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.MEM;*/DE
FLX /RS=MT: 29,1 EDITLB.MEM/DO
TIME
RUN FILSR2
LBR LI/CR:100.:100.:100.:98.1
F4P @LIB
@STUFF
PIP EDITLB.OLB=11.OLB/RF
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.MEM;*/DE
FLX /RS=MT: 29,1 PULFLB.MEM/DO
TIME
RUN FILSR2
LBR LI/CR:290.:100.:100.:98.1
F4P @LIB
@STUFF
PIP PULFLB.OLB=11.OLB/RF
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
```

75

```
PIP LIB.CMD;*/DE
PIP STIFF.CMD;*/DE
PIP *.NEU;*/DE
FLX /RS=HT: 20,1 STIFF.NEU/RO
TIME
RUN FILSP2
LBR L1/CR:170.:190.:199.:OBJ
F4P OLIB
QSTIFF
PIP STIFF.OLB=L1.OLB/RE
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEU;*/DE
FLX /RS=HT: 20,1 BULRB.NEU/RO
TIME
RUN FILSP2
LBR L1/CR:450.:190.:199.:OBJ
F4P OLIB
QSTIFF
PIP BULRB.OLB=L1.OLB/RE
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEU;*/DE
FLX /RS=HT: 20,1 REFL.NEU/RO
TIME
RUN FILSP2
LBR L1/CR:110.:190.:199.:OBJ
F4P OLIB
QSTIFF
PIP REFL.OLB=L1.OLB/RE
```

76

```
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEW;/DE
FLX /RS=RT: 20,1 RESULT.NEW/DO
TIME
RUN FILER2
LBR L1/CR:375.:100.:100.:OBJ
F4P @LIB
QSTUFF
PIP RESUL1.OLB=L1.OLB/RF
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEW;/DE
FLX /RS=RT: 20,1 RESOM.NEW/DO
TIME
RUN FILER2
LBR L1/CR:30.:50.:50.:OBJ
F4P @LIB
QSTUFF
PIP RESOM.OLB=L1.OLB/RF
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEW;/DE
FLX /RS=RT: 20,1 RRITE.NEW/DO
TIME
RUN FILER2
LBR L1/CR:300.:100.::OBJ
F4P @LIB
```

```
@STUFF
PIP EDIT*.OLB=LL.OLB/PU
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEW;*/DE
FLX /RS=MT: 29,1 DEFCS.NEW/DC
TIME
RUN FILCP2
LBR LL/CR:90.:50.:OBJ
F&R @LIB
@STUFF
PIP DEFCS.OLB=LL.OLB/PU
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEW;*/DE
FLX /RS=MT: 29,1 LOCAL.NEW/DC
TIME
RUN FILCP2
LBR LL/CR:90.:50.:OBJ
F&R @LIB
@STUFF
PIP LOCAL.OLB=LL.OLB/PU
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.NEW;*/DE
FLX /RS=MT: 29,1 CUPS.NEW/DC
TIME
RUN FILCP2
```

```
LBR 11/CR:70.:50.::CRI
FDP OLIB
@STUFF
PIP SUBS.OLB=11.OLB/RP
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.MEM;*/DE
FIX /RS=RT: 20,1 TRANS.MEM/RO
TIME
RUN FILES2
LBR 11/CR:75.:40.::CRI
FDP OLIB
@STUFF
PIP TRANS.OLB=11.OLB/RP
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.MEM;*/DE
FIX /RS=RT: 22,1 OPTIM.MEM/RO
TIME
RUN FILES2
LBR 11/CR:45.:40.::CRI
FDP OLIB
@STUFF
PIP OPTIM.OLB=11.OLB/RP
PIP *.FTN;*/DE
PIP *.OBJ;*/DE
PIP LIB.CMD;*/DE
PIP STUFF.CMD;*/DE
PIP *.MEM;*/DE
FIX /RS=RT: 20,1 PROCS.MEM/RO
```

```
TIME
RUN FILSP2
LBR L1/CR:90.:160.::031
F4P SLIB
QSTIFF
PIP PEOCS.OLB=11.OLB/RE
PIP *.OBJ;*/DE
PIP *.FTN;*/DE
PIP LIB.CMD;*/DE
PIP QSTIFF.CMD;*/DE
F4X /RS=MT: 20,1 SAVER.MEU/DO
F4X /RS=MT: 20,1 REGIEU.MEU/DO
F4X /RS=MT: 20,1 QILKE.MEU/DO
F4X /RS=MT: 20,1 QUTOL.MEU/DO
F4X /RS=MT: 20,1 TRAI1.MEU/DO
F4X /RS=MT: 20,1 QRIIT.MEU/DO
F4X /RS=MT: 20,1 TRAI2.MEU/DO
F4P QAVER=QAVEK.MEU/CO:20
F4P REGIEU=REGIEU[DL,MEU]/CO:20
F4P QILKE=QILKE.MEU/CO:20
F4P QUTOL=QUTOL.MEU/CO:20
F4P TRAI1=TRAI1.MEU/CO:20
F4P QRIIT=QRIIT.MEU/CO:20
F4P TRAI2=TRAI2.MEU/CO:20
PIP *.MEU;*/DE
F4X /RS=MT: 20,1 EST.TSK/DO
RUN EST
F4X /RS=MT: 20,1 EST05.I05/DO
PIP *.TSK;*/DE
PIP *.DAT;*/DE
QILBES
PIP *.OBJ;*/DE
PIP *.CMD;*/DE
PIP *.QEL;*/DE
```

APPENDIX F

LISTING OF GIFTS TAPES
(NBS VERSION)

The following are listings of the contents of the
tapes needed by BUILDT.CMD to build GIFTS. Though all
the files could have fit on one tape, the method of dividing
them was used to make the reading process more efficient.
In general, the source listings are on MK0: (MT:) and the
CMD, ODL and existing TSK files are on MT1:.

The files were created under the FLX utility of RSX-11M
in DOS format.

```
DIRECTORY        MTO:[20,1]
01-SEP-80

LIBR1.NEW                 125.    01-SEP-80
LIBR2.NEW                 172.    01-SEP-80
LIBR3.NEW                 172.    01-SEP-80
LIBR4.NEW                 260.    01-SEP-80
LIBR5.PDP                 166.    01-SEP-80
BULKLB.NEW                351.    01-SEP-80
BULKM.NEW                 577.    01-SEP-80
EDITM.NEW                 449.    01-SEP-80
BULKF.NEW                  20.    01-SEP-80
EDITLB.NEW                263.    01-SEP-80
STIFF.NEW                 164.    01-SEP-80
DECOM.NEW                  23.    01-SEP-80
STRESS.NEW                229.    01-SEP-80
AUTOL.NEW                  26.    01-SEP-80
RESULT.NEW                544.    01-SEP-80
TRAN1.NEW                  24.    01-SEP-80
TRAN2.NEW                  26.    01-SEP-80
REDCS.NEW                 104.    01-SEP-80
LOCAL.NEW                 120.    01-SEP-80
SAVER.NEW                   8.    01-SEP-80
RESIDU.NEW                 20.    01-SEP-80
PRINT.NEW                  20.    01-SEP-80
TEST.NEW                    2.    01-SEP-80
BULKS.NEW                 646.    01-SEP-80
LOADS.NEW                 551.    01-SEP-80
OPTIM.NEW                  52.    01-SEP-80
DEFL.NEW                  115.    01-SEP-80
TRANS.NEW                  91.    01-SEP-80
DEFCS.NEW                 152.    01-SEP-80
TEST2.NEW                   3.    01-SEP-80
TSTPLT.NEW                  6.    01-SEP-80
SUBS.NEW                   52.    01-SEP-80

TOTAL OF 5533. BLOCKS IN 32. FILES
```

```
DIRECTORY       MT1:[20,1]
01-SEP-80

FILSR2.TSK            57.       01-SEP-80
EST.TSK              56.       01-SEP-80
STUFFE.TSK           41.       01-SEP-80
BUILD.CMD             9.       01-SEP-80
BUILDT.CMD           10.       01-SEP-80
EST.CMD               1.       01-SEP-80
STIFF.CMD             1.       01-SEP-80
SUBS.CMD              1.       01-SEP-80
TRAN1.CMD             1.       01-SEP-80
TRANS.CMD             1.       01-SEP-80
OPTIM.CMD             1.       01-SEP-80
STRESS.CMD            1.       01-SEP-80
EDITLB.CMD            1.       01-SEP-80
SAVEK.CMD             1.       01-SEP-80
PRINT.CMD             1.       01-SEP-80
RESIDU.CMD            1.       01-SEP-80
TRAN2.CMD             1.       01-SEP-80
REDCS.CMD             1.       01-SEP-80
GIFTS5.CMD            1.       01-SEP-80
BULKM.CMD             1.       01-SEP-80
BULKLB.CMD            1.       01-SEP-80
DEFL.CMD              1.       01-SEP-80
RESULT.CMD            1.       01-SEP-80
BULKF.CMD             1.       01-SEP-80
EDITM.CMD             1.       01-SEP-80
AUTOL.CMD             1.       01-SEP-80
DECOM.CMD             1.       01-SEP-80
DEFCS.CMD             1.       01-SEP-80
LOCAL.CMD             1.       01-SEP-80
BUILDD.CMD           10.       01-SEP-80
DECOM.ODL             1.       01-SEP-80
OPTIM.ODL             1.       01-SEP-80
BULKM.ODL             3.       01-SEP-80
BULKLB.ODL            2.       01-SEP-80
DEFL.ODL              2.       01-SEP-80
EDITM.ODL             2.       01-SEP-80
LOCAL.ODL             1.       01-SEP-80
SUBS.ODL              1.       01-SEP-80
DEFCS.ODL             1.       01-SEP-80
TRANS.ODL             1.       01-SEP-80
STRESS.ODL            1.       01-SEP-80
STIFF.ODL             2.       01-SEP-80
REDCS.ODL             1.       01-SEP-80
RESULT.ODL            3.       01-SEP-80
EDITLB.ODL            1.       01-SEP-80
GIFTS5.INF          197.       01-SEP-80
EST.FTN               2.       01-SEP-80
FILSOR.FTN           21.       01-SEP-80
FILSR2.FTN           13.       01-SEP-80
STUFFE.FTN            6.       01-SEP-80
FILSOR.TSK           51.       01-SEP-80

TOTAL OF 520. BLOCKS IN 51. FILES
```

84

BIBLIOGRAPHY

Digital Equipment Corporation, FORTRAN Four-Plus, User's Guide, 1977.

Digital Equipment Corporation, PDP-11 FORTRAN, Language Reference Guide, 1977.

Digital Equipment Corporation, RSX-11M, Beginner's Guide, 1977.

Digital Equipment Corporation, RSX-11M, Task Builder Reference Manual, 1978.

Eckhouse, R. H. Jr., Morris, L. R., Minicomputer Systems, Organization, Programming and Applications (PDP-11), 2d Ed., Prentice-Hall, 1979.

Rogers, D. F., Adams, J. A., Mathematical Elements for Computer Graphics, McGraw-Hill, 1976.

Zienkiewicz, O. C., The Finite Element Method, 3d Ed., McGraw-Hill, 1977.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor Gilles Cantin, Code 69CI Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	5
5.	Professor Hussein A. Hamel University of Arizona College of Engineering Aerospace and Mechanical Engineering Department Interactive Graphics Engineering Laboratory AME Building, Room 210 Tucson, Arizona 85721	2
6.	LCDR John T. Sheldon, USN 218 Camino Entrada Chula Vista, California 92010	1
7.	Professor George A. Rahe, Code 52Ra Department of Computer Science Naval Postgraduate School Monterey, California 93940	1

DATE
FILMED

8